## Obstacles Overcame

One of the obstacles I overcame was how to use the operators to compare strings directly. When writing positionOfMax, at first I compared the strings using a[i].length(), thinking by “max” the program is asking for the “longest” string. Fixing this is rather easy: just substitute it with *if(a[i]>a[pos])*, using > to compare them directly.

Another obstacle I overcame was considering special cases for countRuns. I figured out the method of counting the times the element changed:

*if(a[i]!=a[i+1]) cnt++;*

If the elements changed once, the answer should be 2 because that transition connects two different runs. Hence the counter should plus 1.

However, when n=0, there will be no runs in the array because there are no elements in question, so the answer should be 0 while my program returned 1. In order to solve this problem, I added a line to address the situation of n==0 and return 0 immediately for that case.

Another obstacle I overcame was the splitters function. Without using extra arrays, this function is one of the harder ones. Bubble sort came to my mind at first, but I always tend to avoid bubble sorts due to its time complexity. So I went for the quick sort method: rearranging the array that the elements smaller than the splitter are on the left side.

*int i=0;*

*for(int j=0;j<n;j++)*

*if(a[j]<splitter)*

*swap(a[j],a[i++]);*

However, this has one problem that for a[j] == splitter, no actions are taken and they will remain at their positions, probably after those > splitter. In order to fix this, I ran another loop after this to pick all a[j] == splitter and swap them to the middle, so that all those a[j] > splitter come after all the other elements.

*int temp=i;*

*for(int j=i;j<n;j++)*

*if(a[j]==splitter)*

*swap(a[j],a[temp++]);*

## Test Data List

| Reason | array | function |
| --- | --- | --- |
| test appendToAll | "rishi", "margaret", "liz", "theresa" | appendToAll(g,4,"?") |
| test lookup | "rishi", "margaret", "gordon", "tony", "", "john", "liz" | lookup(h,7,”john”) == 5 |
| test lookup:  small n / not found | same as above | lookup(h,2,“gordon”)==-1 |
| test positionOfMax | same as above | positionOfMax(h,7) == 3 |
| test positionOfMax | "david", "liz", "margaret", "tony", "gordon", "boris" | positionOfMax(pm,6) == 3 |
| test rotateLeft | "rishi", "margaret", "liz", "theresa" | rotateLeft(g,4,1) |
| test rotateLeft:  pos>n-1 | same as above | rotateLeft(g,4,4) == -1 |
| test countRuns | "margaret", "margaret", "margaret", "tony", "tony" | countRuns(d, 5) == 2 |
| test countRuns:  n=0 | same as above | countRuns(d, 0) == 0 |
| test countRuns:  n=1 | same as above | countRuns(d, 1) == 1 |
| test flip:  odd n | "liz", "gordon", "tony" | flip(f, 3) |
| test flip:  partly flipped | "rishi", "margaret", "gordon", "tony", "", "john", "liz" | flip(h, 4) |
| test differ | h[7] = { "rishi", "margaret", "gordon", "tony", "", "john", "liz" };  g[4] = { "rishi", "margaret", "liz", "theresa" }; | differ(h, 4, g, 4) == 2 |
| test differ:  runs out when equal | same as above | differ(h, 2, g, 1) == 1 |
| test subsequence | h[7] = { "rishi", "margaret", "gordon", "tony", "", "john", "liz" };  e[4] = { "gordon", "tony", "", "john" }; | subsequence(h,7,e,4)== 2 |
| test subsequence:  not found | h[7] = { "rishi", "margaret", "gordon", "tony", "", "john", "liz" };  e[4] = { "gordon", “kanye", "", "john" }; | subsequence(h,7,e,4)==-1 |
| test lookupAny | h[7] = { "rishi", "margaret", "gordon", "tony", "", "john", "liz" };  f[3] = { "liz", "gordon", "tony" }; | lookupAny(h,7,f,3)== 2 |
| test lookupAny:  not found | h[7] = { "rishi", "margaret", "gordon", "tony", "", "john", "liz" };  f[3] = { "tyler", "kanye", “kendrick" }; | lookupAny(h,7,f,3)== -1 |
| test split | "david", "liz", "margaret", "tony", "gordon", "boris" | split(pm,6,”john") == 3 |
| test split:  a[i]==splitter | "rishi", "margaret", "gordon", "tony", "", "john", "liz" | split(h, 7, "liz") == 3 |
| test split:  multiple a[i]  == splitter | "margaret", "theresa", "liz", "rishi", "rishi" | split(pm2,5,”rishi")== 2 |